Soft-Decision Reed-Solomon Decoding on Partial Response Channels

Michael K. Cheng*, Jorge Campello†, and Paul H. Siegel*
*Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407
† IBM Almaden Research Center, San Jose, CA 95120

***Abstract*— We compare the performance of list-type soft-decision Reed-Solomon (RS) decoding algorithms to that of classical hard-decision RS decoding on Partial Response (PR) channels. The two soft-decision RS decoding approaches that we consider, the List-GMD and the Koetter-Vardy algorithm, are both based on Sudan's (polynomial) interpolation approach to polynomial-time list-decoding. The soft-decision RS decoders take as input symbol-based reliabilities which can be provided by a symbol-based BCJR algorithm. The symbol-based BCJR algorithm is an extension of the original BCJR algorithm and calculates the *a posteriori probability* (APP) of a block of *l*-bits. The complexity of the algorithm can be reduced when applied to a PR channel. We present a hybrid Viterbi-BCJR approach that can be used when only the reliability of the most-likely symbol is desired. The hybrid approach calculates the APP of the most reliable symbol without the need to run the complete forward and backward algorithm. We demonstrate through simulations that soft-decision RS decoding will lead to a lower probability of decoding error. Moreover, we show that using the symbol-based APPs will yield a lower symbol error rate (SER) than using the measures obtained by multiplying the bit reliabilities.**

I. Introduction

We consider a discrete memoryless source that is transmitted over a channel affected by intersymbol interference (ISI) and additive white Gaussian noise (AWGN.) A specific scenario is the magnetic recording channel. To combat ISI in this channel, we typically use an equalizer to modify the channel read-back signal into a pre-determined partial response (PR) target [1, Chapter 9]. The equalized magnetic recording channel is also known as a PR channel and specified by a PR polynomial. For maximum likelihood detection, we can use the Viterbi algorithm applied to the trellis corresponding to the PR polynomial.

In a magnetic recording system, we often employ a concatenated coding scheme where the "inner code" is the PR channel and the outer code is an algebraic block code. Partial Response Maximum Likelihood (PRML) is a detection technique used to select the most-likely bit-sequence out of the inner channel and an algebraic outer code such as the Reed-Solomon (RS) code is used for the residual errors.

The classical bounded-distance RS decoder correctly decodes to a unique codeword if the number of errors t is less than half the minimum distance d. To exceed the classical decoding radius t, "list-decoding" was introduced independently by Elias and Wozencraft [2], [3]. List-decoding is a technique that, for a given received vector $\underline{v}$, efficiently generates a list of codewords within a Hamming distance τ from $\underline{v}$, where τ is greater than or equal to t. Sudan [4] and Guruswami [5] were the first to develop an approach that solves the list-decoding problem in polynomial time. Their approach to list-decoding consists of polynomial interpolation and factorization. In Section II, we present the List-GMD algorithm which combines Sudan and Guruswami's list-decoding with Forney's [6] Generalized Minimum Distance (GMD) decoding into a sequential erasure list-decoding attempt at soft-decision Reed-Solomon decoding. Koetter and Vardy [7] have also developed a soft-decision algebraic decoding algorithm for RS codes based on Sudan and Guruswami's polynomial interpolation. Both the List-GMD and the Koetter-Vardy algorithm take as inputs symbol-wise reliability information (or APPs) from the inner PR channel. One way to approximate the symbol-wise APPs is to apply the original BCJR algorithm [8] to the inner channel and multiply the bit APPs that form a symbol. However, in using the bit-product approach, we may include invalid trellis paths in the calculation. In general, to get reliabilities for l-bit symbols we are interested in $Pr\left(u_{k-(l-1)}^{k}=\varphi \mid R_1^N\right)$, $\forall\varphi\in\{0,1\}^l$, and for all $k=il$, $i=1,2,\cdots$. This could be done by first calculating $Pr\left(u_1^N \mid R_1^N\right)$ and then obtaining the l-dimensional marginals. It is, however, infeasible to compute the conditional distribution $Pr\left(u_1^N \mid R_1^N\right)$ because the number of points in the probability distribution is exponential in N. The symbol-based BCJR, much like the bit-based BCJR, uses the Markov properties of the source to get around the exponential complexity of calculating the l-dimensional marginals. Note that we cannot compute the l-dimensional marginals from the product of the single-bit marginals and, therefore, we cannot multiply the bit APPs to calculate the symbol APPs. To correctly calculate the l-dimensional marginals and, thus the symbol-wise APPs, we discuss a symbol-based BCJR algorithm in Section III. Our method uses the conventional bit-based trellis as in [8] and only modifies the manner in which the probability functions are updated when compared to the original BCJR algorithm. Hoeher [9] was the first to develop a method of calculating the *a posteriori probability* of a block of l consecutive bits. We however provide a description and derivation that is aimed at applications in the magnetic recording channel. We show that simplifications of the algorithm can be made for the case of binary-input ISI channels. We also extend Hoeher's work by introducing a reduced-complexity hybrid Viterbi-BCJR algorithm that calculates the reliability of the most-likely symbol. In Section IV, we provide simulation results that demonstrate the performance improvement of soft-decision RS decoding over hard-decision decoding and the reduction in SER that arises from using the APPs calculated by the symbol-based BCJR algorithm. In Section V we conclude with observations and remarks.

II. Soft-decision Reed-Solomon decoding

A. GMD decoding

Decoding using soft information improves performance. Forney's Generalized Minimum Distance (GMD) decoding [6, Chapter 3] was an early attempt at using soft channel information in the decision process. GMD decoding takes as inputs the quantized received vector $\underline{v}=\{v_1,\cdots,v_n\}$ and its associated reliability vector $\underline{r}=\{r_1,\cdots,r_n\}$, where $0\le r_i\le 1$, $i=1,\cdots,n$. The algorithm performs a series of erasures-and-errors hard-decision decoding on $\underline{v}$ by erasing the s least reliable symbols according to $\underline{r}$. If a codeword satisfies the GMD criterion [6, Eq. 3.41], it will be the unique codeword that does so and be found by the algorithm.

This work was supported by the National Storage Industry Consortium and by the Center for Magnetic Recording Research, UCSD.

B. List-GMD decoding

List-GMD decoding is a combination of Sudan's list-decoding and Forney's GMD decoding. The algorithm takes as inputs a received vector $\underline{v}$ and its associated reliability vector $\underline{r}$ which can be calculated by, for example, the hybrid Viterbi-BCJR algorithm to be discussed in Section III. Sort the reliabilities in order of increasing magnitude, that is, $r_{i_1} \leq \cdots \leq r_{i_n}$. Define the indicator vector as $\mathbf{q}_s = \{q_s(r_1), \cdots, q_s(r_n)\}$, where $0 \leq s \leq n$ and

$$q_s\left(r_{i_j}\right) = \begin{cases} 0, & 1 \leq j \leq s \\ 1, & s+1 \leq j \leq n \end{cases} \qquad (1)$$

The **List-GMD algorithm** is as follows:

1) *For $s = 0, \cdots, d-1$ do*
   a) *for each zero position in $\mathbf{q}_s$, erase the corresponding symbol in $\underline{v}$. Denote this vector with s erased symbols $\underline{v}_s$.*
   b) *perform erasures-and-errors* ***list-decoding*** *on $\underline{v}_s$ thus generating a list of candidate codewords for each decoding trial.*
2) *Select the most likely codeword from the union of the lists output by the decoding trials.*

For an (n, k, d) Reed-Solomon code with s erasures, the error correction bound of Sudan's erasures-and-errors decoding is given by [5, Theorem 16]:

$$\tau(s) < (n-s) - \sqrt{n(k-1)}; \qquad (2)$$

that is, we can correct up to $\tau(s)$ errors when we apply erasures-and-errors list-decoding to a received vector $\underline{v}$ with s symbols erased. List-GMD decoding will perform a series of list-decodings and attempt to correct up to $\tau(s)$ errors and fill in s erasures for $s \in [0, \cdots, d-1]$.

We use Example 1 in Koetter and Vardy's paper [10] to illustrate List-GMD decoding. The transmitted codeword is $\{1,2,3,4,0\} \in \mathbb{C}_5(5,2)$, an extended Reed-Solomon code over $GF(5)$. In this example, applying maximum likelihood (ML) decoding will lead to the transmitted codeword. The hard-decision received vector is $\{4,2,3,3,3\}$, where the first, fourth, and fifth symbols are incorrect. The corresponding reliability vector is $\{0.90, 0.99, 0.61, 0.44, 0.40\}$. A codeword $\underline{c}$ that has $d_H(\underline{c}, \underline{v}) \leq \tau$ is τ-consistent.

Example 1: List-GMD decoding

- Decoding $\{4,2,3,3,3\}$ using Sudan's list-decoding algorithm, we obtain a list of 2-consistent, $\tau(0) = 2$, codewords in the RS code: $\{3,3,3,3,3\}$ and $\{4,2,0,3,1\}$.
- Decoding $\{4,2,3,3,?\}$, where ? indicates the erased positions, we obtain a list of 1-consistent, $\tau(1) = 1$, codewords: $\{4,2,0,3,1\}$.
- Decoding $\{4,2,3,?,?\}$, we obtain a list of 1-consistent, $\tau(2) = 1$, codewords: $\{1,2,3,4,0\}$, $\{4,1,3,0,2\}$, and $\{4,2,0,3,1\}$.
- Decoding $\{4,2,?,?,?\}$, we obtain a list of 0-consistent, $\tau(3) = 0$, codewords: $\{4,2,0,3,1\}$.

List-GMD decoding $\{4,2,3,3,3\}$ therefore gives us the following codewords: $\{3,3,3,3,3\}$, $\{4,2,0,3,1\}$, $\{1,2,3,4,0\}$, and $\{4,1,3,0,2\}$. In this case List-GMD was able to find the maximum likelihood codeword $\{1,2,3,4,0\}$, whereas classical and GMD decoding cannot. Notice that we do not have to repeat the list-decoding trials for every $s \in [0, \cdots, d-1]$. The output list generated by decoding $\{4,2,3,?,?\}$ contains the output list produced by decoding $\{4,2,3,3,?\}$. We can skip unnecessary List-GMD trials by using the relations given in an errors-erasures table, such as the one generated for the example, Table I. According to Table I, the 1-erasure case and 2-erasures case can correct up to $\tau = 1$ error. The list of codewords generated by the 2-erasures case will contain all codewords produced by the 1-erasure case, because the two cases differ by an erasure which can always be filled-in correctly. To run List-GMD on a received vector, we only have to run the algorithm once for every value of $\tau(s)$. For the cases where multiple s evaluate to the same $\tau(s)$, we only have to list-decode the case with the largest $\tau(s) + s$ combination. Moreover, the errors-erasures table only depends on n, k, and s, so we would only have to generate the table once for each code before we begin List-GMD decoding.

TABLE I
ERRORS-ERASURES TABLE FOR THE $(5,2,4)$ REED-SOLOMON CODE

s	0	1	2	3
$\tau(s)$	2	1	1	0

C. The Koetter-Vardy algebraic soft-decision decoding

Koetter and Vardy [7], [10] developed a polynomial-time soft-decision decoding algorithm based on Sudan's list-decoding. Koetter and Vardy's approach uses polynomial interpolation with variable multiplicities while Sudan's technique uses polynomial interpolation with fixed multiplicities. For an (n, k, d) RS code defined over $GF(q)$, the Koetter-Vardy (K-V) algorithm generates a size $q \times n$ multiplicity matrix $M = \{m_{i,j}\}$, $i = 1, \cdots, q$ and $j = 1, \cdots, n$, from channel posterior probabilities for a maximum possible $q \cdot n$ interpolation points. The allocation of multiplicities in the $q \times n$ matrix M is done by a greedy algorithm [10, Algorithm A]. Each entry in M can be a different non-negative integer. Sudan's list-decoding can be viewed as a special case of the K-V algorithm with a multiplicity matrix M that consists of one and only one nonzero entry in each column and each entry has the same value. The K-V approach allows the more reliable entries in M to receive higher multiplicity values and this yields the potential for improved performance.

The complexity of the K-V algorithm depends on the cost of the multiplicity matrix, defined by Koetter and Vardy [10] as

$$\mathcal{C}(M) \triangleq \frac{1}{2} \sum_{i=1}^{q} \sum_{j=1}^{n} m_{i,j}\left(m_{i,j}+1\right). \qquad (3)$$

Let $\mathcal{C} = \mathcal{C}(M)$. The computation of the interpolating polynomial $Q_M(X,Y)$ is equivalent to solving $\mathcal{C}$ linear equations. A straightforward method of solving for $Q_M(X,Y)$ is Gaussian Elimination, however, its complexity is on the order of $O\left(\mathcal{C}^3\right)$. Nielsen [11] has proposed a reduced-complexity method of finding $Q_M(X,Y)$ that is on the order of $O\left(\kappa\mathcal{C}^2\right)$ where κ is a constant. The computational complexity of Koetter-Vardy's algebraic soft-decision decoding can therefore be high. Thus, we would like to use a threshold condition to estimate its performance.

Koetter and Vardy provided a threshold condition for simulation in their paper [10, Corollary 5]. Given a codeword $\underline{c}$, define $[\underline{c}]$ as a $q \times n$ matrix. Let each row index of $[\underline{c}]$ represent an element $\zeta_i \in GF(q)$; then $[\underline{c}]_{i,j} = 1$ if $c_j = \zeta_i$ and $[\underline{c}]_{i,j} = 0$ otherwise. Define the score as

$$\mathcal{S}_M\left(\underline{c}\right) = \langle M, [\underline{c}]\rangle = \sum_{i=1}^{q}\sum_{j=1}^{n} m_{i,j}\cdot [\underline{c}]_{i,j}\,. \tag{4}$$

Koetter and Vardy proved that $Q_M\left(X,Y\right)$ has a factor $Y - f\left(X\right)$, where $f\left(X\right)$ evaluates to $\underline{c} \in \mathbb{C}_q\left(n,k\right)$, if

$$\mathcal{S}_M\left(\underline{c}\right) \geq \sqrt{2\left(k-1\right)\mathcal{C}}. \tag{5}$$

The threshold condition of (5) is not tight; that is, codewords that do not satisfy (5) may still be in the K-V output list.

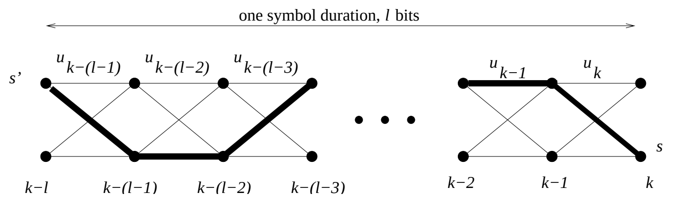

Fig. 1. A simple 2-state trellis to illustrate the indices used in the symbol-based APP derivations. The input bit sequence $\varphi \triangleq \left(b_{l-1}, b_{l-2}, \cdots, b_0\right)$ is marked by the highlighted path.

III. A MODIFIED BCJR ALGORITHM FOR NON-BINARY SYMBOLS

A. Description and derivation

Let $\varphi \in GF(2^l)$ be an information symbol. There are l bits per symbol and we can map each symbol in $GF(2^l)$ to a distinct bit pattern; that is, $\varphi \triangleq \left(b_{l-1}, b_{l-2}, \cdots, b_0\right)$, where $b_i \in GF(2)$. Let $\underline{u} \triangleq u_{k-(l-1)}^{k}$ be the information symbol mapped to the input bit sequence from time $k-(l-1)$ to time k. Figure 1 illustrates the index labeling. The *a posteriori probability* that the information symbol $\underline{u}$ equals φ conditioned on the length-N received bit sequence R_1^N is:

$$Pr\left(\underline{u} = \varphi \mid R_1^N\right) = \frac{p\left(\underline{u}=\varphi, R_1^N\right)}{p\left(R_1^N\right)} \tag{6}$$

$$= \frac{1}{p\left(R_1^N\right)} \sum_s \sum_{s'} p\left(\underline{u} = \varphi, s_k = s, s_{k-l} = s', R_1^N\right) \tag{7}$$

The equivalence in (6) is obtained using Bayes' rule and (7) is obtained using the principle of total probability. By applying Bayes' rule and the Markov property that events after time k only depend on the current state s_k and are independent of past observations, we can rewrite the joint pdf in (7) as

$$\begin{aligned} p\left(\underline{u} = \varphi, s_k = s, s_{k-l} = s', R_1^N\right) &= p\left(R_{k+1}^N \mid s_k = s\right) \\ &\cdot p\left(\underline{u} = \varphi, s_k = s, R_{k-(l-1)}^k \mid s_{k-l} = s'\right) \\ &\cdot p\left(s_{k-l} = s', R_1^{k-l}\right) \qquad (8) \\ &= \beta_k(s)\cdot \gamma_{(k-(l-1),k)}^{\varphi}(s',s)\cdot \alpha_{k-l}(s') \qquad (9) \end{aligned}$$

The term $\beta_k(s)$ is called the backward state metric, the term $\gamma_{(k-(l-1),k)}^{\varphi}(s',s)$ is called the branch transition probability, and the term $\alpha_{k-1}(s')$ is called the forward state metric.

We expand the branch transition probability, by using Bayes' rule and the fact that the symbol *a priori* probability is state-independent, as

$$\begin{aligned} \gamma_{(k-(l-1),k)}^{\varphi}\left(s',s\right) &= Pr\left(\underline{u} = \varphi\right) \\ &\cdot Pr\left(s_k = s \mid \underline{u} = \varphi, s_{k-l} = s'\right) \\ &\cdot p\left(R_{k-(l-1)}^k \mid s_k = s, \underline{u} = \varphi, s_{k-l} = s'\right). \qquad (10) \end{aligned}$$

If state s at time k is connected to state s' at time $k-l$ via the input sequence $\underline{u} = \varphi$, then $Pr\left(s_k = s \mid \underline{u} = \varphi, s_{k-l} = s'\right) = 1$; otherwise $Pr\left(s_k = s \mid \underline{u} = \varphi, s_{k-l} = s'\right) = 0$. The pdf $p\left(R_{k-(l-1)}^k \mid s_k = s, \underline{u} = \varphi, s_{k-l} = s'\right)$ is a function of the channel characteristic; in a partial response channel with AWGN the pdf can be calculated as:

$$\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^l e^{\frac{-\sum_{i=0}^{l-1}\left(R_{k-i}-c_i\right)^2}{2\sigma^2}} \tag{11}$$

where σ^2 is the noise variance and $\left(c_{l-1}, c_{l-2}, \cdots, c_0\right)$ is the partial response channel output sequence that corresponds to the input sequence $\varphi \triangleq \left(b_{l-1}, b_{l-2}, \cdots, b_0\right)$. The forward state metric $\alpha_k(s)$ and the backward state metric $\beta_s\left(s\right)$ can be updated as in the original BCJR [8].

B. Simplification for ISI channels

There are simplifications that can be made for the case of binary-input ISI channels in order to reduce complexity. The simplifications come from the fact that the states represent subsequences of the input sequence. For a channel with memory ν, we can calculate the joint pdf as:

$$\begin{aligned} & p\left(\underline{u} = \varphi, s_k = s, s_{k-l} = s', R_1^N\right) \\ &= p\left(\underline{u} = \left(b_{l-(\nu+1)}, \cdots, b_0\right), s_k = s, s_{k-(l-\nu)} = s'', R_1^N\right) \\ &\qquad = \alpha_{k-(l-\nu)}\left(s''\right)\gamma_{(k-(l-(\nu+1)),k)}^{\left(b_{l-(\nu+1)},\cdots,b_0\right)}\left(s'',s\right)\beta_k\left(s\right) \qquad (12) \end{aligned}$$

where s'' is the state that corresponds to the shift register configuration after an input of ν bits. Applying the simplification would only require $N_{ISI} = 2 + (l-\nu)$ operations as opposed to $N_{general} = (2+l)\,2^\nu + (2^\nu - 1)$ operations.

C. The Hybrid Viterbi-BCJR algorithm

If we are only interested in the most-likely l-bit symbol, we can apply a variation of the symbol-based BCJR to further save on complexity. The details of the algorithm are described below. It is assumed that we have a trellis for an ISI channel with memory ν; that is, there are 2^ν states that are in one-to-one correspondence with all 2^ν possible binary strings of length ν.

1) The BCJR Phase : Run the BCJR algorithm storing the forward metrics α at the time instances $k-(l-\nu)$ and the backward metrics β at time instances k, where $k = il$, $i = 0, 1, \cdots, N_s$ and N_s is the number of symbols to be detected.

2) The Viterbi Phase: **Initialization:** The backwards Viterbi metrics at time k, μ_k, are initialized with the BCJR β's at time k; that is, $\mu_k\left(S_i\right) = \beta_i^k$, $i = 0, \cdots, 2^\nu - 1$. **Propagation:** For each state $s' \in \mathcal{S}$ at time $j = k-1, \cdots, k-(l-\nu)$: (a) Calculate the accumulated branch metric for the two edges connecting state s' at time j to a state at time $j+1$. Let the two states at time $j+1$ that connect to s' be s_0 and s_1 corresponding respectively to the edges with labels 0 and 1. The accumulated branch metrics are given by $\tilde{\mu}_j^b\left(s'\right) = \gamma_{(j,j+1)}^b\left(s', s_b\right)\mu_{j+1}\left(s_b\right),\ b = 0, 1.$ (b) Compare the two accumulated branch metrics and select the largest. Let us say that the selected branch has label b^*. (c) Update the state metric and the survivor sequence q. In $q_j\left(s\right)$ we store the state at time $j+1$ corresponding to the sequence with largest metric starting at time k (with metrics given by the β's) and ending at state s at time j. The update equations are $\mu_j\left(s'\right) = \tilde{\mu}_{j+1}^{b^*}\left(s'\right)$ and $q_j\left(s'\right) = s_{b^*}$.

3) Termination: For each state at time $k-(l-\nu)$, calculate the overall state metrics, $\lambda(S_i)$, as $\lambda(S_i) = \mu_{k-(l-\nu)}(S_i)\,\alpha_i$. Find the state s^* with the largest overall metric. The most-likely l-bit symbol, φ^*, made up of bits from positions $k-l+1$ to k is given by the ν bits corresponding to state s^* followed by the $l-\nu$ bits obtained by reading off the edge labels from the survivor sequence $q_{k-(l-\nu)}(s^*)$. The probability of the most-likely symbol is given by $Pr\left(\varphi^* \mid R_1^N\right) = \lambda(s^*)/p\left(R_1^N\right)$ and $p\left(R_1^N\right)$ is obtained from the forward portion of the BCJR algorithm [8].

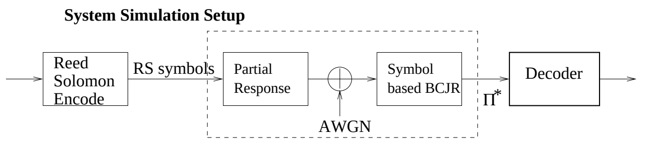

Fig. 2. Partial Response system model to compare various RS decoding techniques.

IV. Simulation Results

A. Performance of Reed-Solomon decoding techniques over a partial response channel

We simulate the performances of Koetter-Vardy (K-V), List-GMD, Sudan, GMD, and the classical bounded-distance Reed-Solomon decoding algorithms on the so-called EPR4 partial response channel with transfer function $h(D) = (1-D)(1+D)^2$. Our system model is shown in Figure 2. It is a simple yet effective model that allows us to compare the various RS decoding techniques. The matrix Π^* is what Koetter and Vardy referred to as the *generalized reliability matrix* [10, Section 10] and applies to channels with memory. We first discuss the performance of Sudan's list-decoding. To evaluate the list-decoding performance, we can run the actual algorithm or simulate the performance of a τ-error correcting code by declaring a successful decode whenever the Hamming distance of the transmitted codeword and the received vector is less than or equal to τ. The simulation assumes that the correct codeword is always selected from the list. Nielsen [11, Ch. 3, Sec. 3] showed that the probability of a list with multiple codewords becomes smaller as the code length n and alphabet size q increase. For code parameters of practical interest in the magnetic recording setting, list-decoding will almost always produce either a one-codeword list or a zero-codeword list. As an example, for a $(255,232,24)$ RS code $Pr\left(|\tau-\text{consistent list}|>1\right) \le 2.2649\times10^{-7}$. Furthermore, the probability of choosing incorrectly from the list of candidates is also small. For these reasons, the simulation results obtained through our assumption will closely approximate the actual decoding performance. Figure 3 compares the performance of GMD, List-GMD, K-V, classical and list decoding. The hybrid Viterbi-BCJR algorithm can be used in place of the symbol-based BCJR algorithm to generate the reliabilities of the most-likely symbols in the cases of GMD and List-GMD decoding. We see that the soft-decision algorithms outperform the classical hard-decision algorithm. List-decoding led to a 0.8 dB gain over classical RS decoding. List-GMD provided about 1.25 dB gain over GMD because we apply sequential list-decoding instead of sequential classical decoding. The performance of the K-V algorithm is affected by the total number of interpolation points used. In our simulation, we use a total of $10n$ interpolation points, where n is the length of the RS code. With only $10n$ points, K-V decoding already achieves the performance provided by List-GMD. Koetter and Vardy have shown that $20n$ interpolation points would provide near-asymptotic (large number of interpolation points) performance [12].

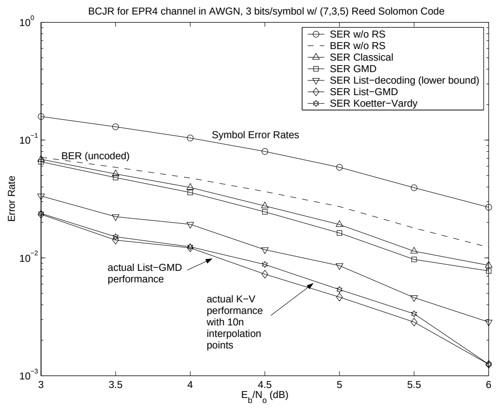

Fig. 3. Comparison of GMD, List-GMD, Koetter-Vardy, classical, and list decoding of a $(7,3,5)$ RS code over $GF\left(2^3\right)$, 3 bits per symbol.

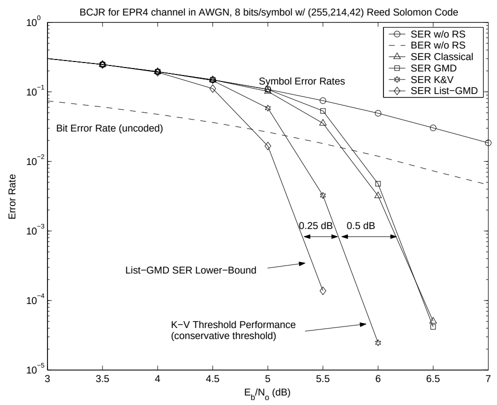

Fig. 4. Threshold performance of List-GMD and Koetter-Vardy decoding of a $(255,214,42)$ RS code over $GF\left(2^8\right)$, 8 bits per symbol. To reduce simulation time, the List-GMD performance is obtained using the lower-bound derived in [13] and the Koetter-Vardy performance is obtained using the threshold condition given in (5).

In Figure 4, we plot the threshold performance of K-V, List-GMD, and classical decoding for a $(255,214,42)$ RS code over $GF(256)$. The List-GMD curve is a calculated lower bound [13], while the K-V curve is obtained using the threshold condition of (5). The threshold K-V curve indicates a 0.5 dB gain and the List-GMD lower bound indicates a 0.75 dB gain over classical RS decoding. In practice, the K-V decoding algorithm will outperform the List-GMD decoding algorithm when a large number of interpolation points is used because the List-GMD algorithm is based on list-decoding and, therefore, does not allow interpolation with variable multiplicities as discussed in Section II-C.

B. Symbol-based BCJR versus bit-product BCJR

We compare the performance of a magnetic recording system that uses the bit-product reliability measures to one that uses the symbol-wise APPs to make hard-decisions on the information symbols. The simulation platform is shown in Figure 5. We compare the decisions ($\widehat{\underline{u}}$) with the transmitted symbols ($\underline{u}$) to calculate the symbol error rate (SER) for each approach. We convert the decisions ($\widehat{\underline{u}}$) and the transmitted symbols ($\underline{u}$) into bits and calculate the bit error rate (BER) for each approach.

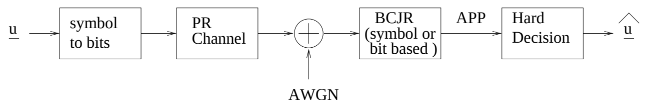

Fig. 5. Simulation setup to compare the symbol-based BCJR to the bit-product BCJR algorithm.

We plot the simulation result generated on the EPR4 channel with 8 bits per symbol in Figure 6. It is interesting to consider the effect that the trellis complexity and the symbol size have on the performance difference between the symbol-based and bit-based BCJR algorithms. If we let S be the number of states per stage and l be the number of bits per symbol, then the number of valid paths per symbol will be S, independent of l, and the total number of possible paths (both valid and invalid) per symbol will be S^l. We therefore would expect the performance improvement in using the symbol-based APPs to increase with increasing trellis complexity S and increasing bits per symbol l [14].

Finally, we apply the reliability information generated by the two approaches to the the K-V decoder. We again refer the readers to [7], [10] for details. For the K-V decoder, we used the simple threshold condition discussed in Section II-C. The simulation output generated by using a $(255, 214)$ Reed-Solomon code on the EPR4 channel is shown in Figure 7. Using reliabilities generated by the symbol-based BCJR algorithm, we found a 0.25 dB SNR gain at a SER of 10^{-3}.

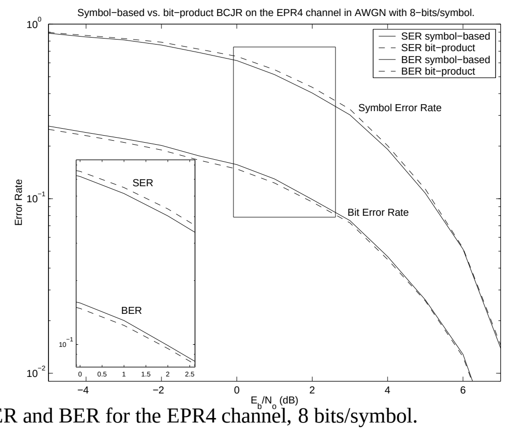

Fig. 6. SER and BER for the EPR4 channel, 8 bits/symbol.

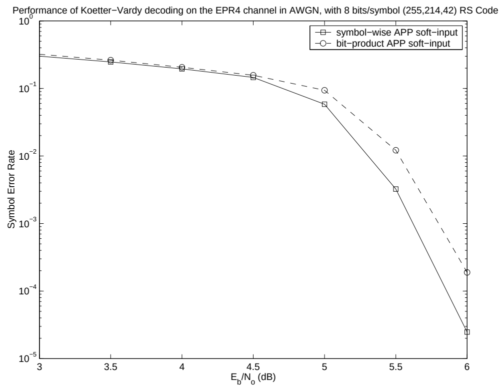

Fig. 7. Performance comparison of K-V decoding using the reliability information produced by the symbol-based BCJR algorithm versus the bit-product approach.

V. Conclusions

We have discussed soft-decision Reed-Solomon decoding algorithms and their application in a magnetic recording channel. The List-GMD and the Koetter-Vardy algorithms can utilize reliability information provided by the channel. The soft-decision decoding algorithms outperform the classical hard-decision decoding algorithm at the cost of added complexity. We can manage the complexity in List-GMD by limiting the decoding radius τ and in Koetter-Vardy by controlling the number of interpolation points.

We presented a symbol-based BCJR algorithm that calculates the symbol-wise APPs for l-bit symbols. Our method is an extension of the original BCJR algorithm and uses the same bit-based trellis. The symbol-based BCJR technique can be used to generate symbol-wise APPs to be used by soft-decision decoding algorithms for algebraic block codes over $GF\left(2^l\right)$. Simplification of the symbol-based BCJR algorithm can be made when applying it to a binary ISI channel. We also introduced the hybrid Viterbi-BCJR algorithm which can be used when only the reliability of the most-likely symbol is desired. Simulations over a partial response channel show that symbol decisions made by using the symbol-wise APPs yield a lower symbol error rate (SER) than the symbol decisions made by using the bit-product reliability measures.

Acknowledgment

Mike Cheng would like to thank Henry Pfister for inspiring the idea that led to the symbol-based BCJR algorithm.

References

[1] A. M. Taratorin, *Characterization of magnetic recording systems: a practical approach*. Guzik Technical Enterprise, 1996.
[2] P. Elias, “List decoding for noisy channels,” tech. rep., MIT, 1957.
[3] J. M. Wozencraft, “List Decoding,” tech. rep., MIT, 1958.
[4] M. Sudan, “Decoding of Reed-Solomon codes beyond the error correction bound,” *J. Complexity*, vol. 13, pp. 180–193, Sept. 1997.
[5] V. Guruswami and M. Sudan, “Improved decoding of Reed-Solomon and Algebraic-Geometry codes,” *IEEE Trans. Inform. Theory*, vol. 45, pp. 1757–1767, Sept. 1999.
[6] G. D. Forney, Jr., *Concatenated Codes*. Cambridge, MA, USA: M.I.T. Press, 1966.
[7] R. Kötter and A. Vardy, “Algebraic soft-decision decoding of Reed-Solomon codes,” in *Proc. IEEE Int. Symp. Information Theory*, (Sorrento, Italy), IEEE, June 2000.
[8] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, March 1974.
[9] P. Hoeher, “Optimal subblock-by-subblock detection,” *IEEE Trans. Commun.*, vol. 43, pp. 714–717, Feb. 1995.
[10] R. Kötter and A. Vardy, “Algebraic soft-decision decoding of Reed-Solomon codes.” preprint, June 2000.
[11] R. R. Nielsen, “Decoding AG-codes beyond half the minimum distance,” Master’s thesis, Danmarks Tekniske Universitet, Copenhagen, Denmark, Aug. 1998.
[12] A. Vardy, “Algebraic soft-decision decoding of Reed-Solomon codes.” ECE 259AN Lecture, UCSD, Nov. 2001. Slides from the talk.
[13] M. K. Cheng, “Report on List-GMD decoding.” Sept. 2001.
[14] M. K. Cheng, J. Campello, and P. H. Siegel, “Soft-decision Reed-Solomon decoding on Partial Response channels.” preprint, Feb. 2002.